GENETIC MUSIC

Mark Hennings and Denise Kettelberger

Background of the Invention

Genetic material identifies the characteristics that are common to living organisims, as well as unique characteristics of every individual plant and animal. The recent acceleration in our knowledge of specific sequences as well as total genomes has also provided tools for rapidly obtaining this information from a small sample of biological material. Numerous products and methods can now take advantage of this pool of knowledge, adapting the genetic material into novel and interesting new products.

Genetic material includes nucleic acid sequences, DNA and RNA, that are arrangements of four different nucleotides, Adenine (A), cytosine (C), thymine (T), and guanine (G) are the building blocks of a DNA sequence. Much of the DNA in an individual genome encodes the amino acids of specific proteins using a triplet codon of nucleotides to code for each specific amino acid, as well as to signal the start and stop of the encoded protein. Non-coding DNA includes regulatory sequences and signals for the processing of the coded messages. The data stored in an individual's genome can be determined by various known sequencing methods, and this data can then be tapped for a variety of purposes, including individual identification, medical diagnosis, and the like.

Summary of the Invention

The present invention takes advantage of the advances in genetic sequence knowledge and provides a creative solution to analysis of sequence data. In the present invention, genetic sequence data is converted into music characteristic of the individual sequence provided. This genetic music is useful in a variety of products, including medical diagnostics, security identification systems, novelty items such as greeting cards, and the like.

Brief Description f the Drawings

Figure 1 is a schematic diagram demonstrating the generation of a music signal from a DNA sequence sample according to one embodiment of the invention.

Figure 2 is a schematic diagram demonstrating the generation of a music signal from a DNA sequence sample according to one embodiment of the invention.

Figure 3 is a schematic diagram demonstrating the generation of a music signal from a DNA sequence sample according to one embodiment of the invention.

Figure 4 is a schematic diagram demonstrating the generation of a music signal from a DNA sequence sample according to one embodiment of the invention.

Detailed Description of the Invention

Definitions:

As used herein, the following terms are intended to have the noted definitions.

"melodic sequence" is a signal containing melodic information.

"harmonic sequence" is a signal containing musical harmonic information.

"chord" is musical information comprising notes of typically three or more intervals.

"transcriber" is an apparatus configured to receive one data signal and transcribe that data signal into a different data signal; in the present invention, the received DNA signal is transcribed into a musical signal.

"generator" is an apparatus configured to generate a data signal; in the present invention, a musical signal is generated.

"decoder" is an apparatus configured to decode a message within a data signal; in the present invention, the received DNA signal is decoded to determine an amino acid signal.

"music signal" is the music product generated from the data stream; in the present invention, the music signal is generated from received DNA sequence that is decoded and may be melodic music, harmonic music, or a combination of these.

"audio waveform" is a digital or analog signal conveying an audio "recording" of the intended music.

"musical command sequence" is a sequence of commands for a generator (such as a synthesizer) to generate music according to the command sequence.

"data carrier" is an apparatus configured and adapted to carry a data signal; in the present invention, a data carrier may be an electronic medium such as a computer, a compact disk (CD), and the like.

"consumer product" is a product purchased and/or used by consumers.

"clinical analyzer" is an apparatus configured to analyze data from patient samples.

Genetic Material

Genetic material is contained within cells in genomic DNA. The nucleotide bases of genomic DNA (A, C, T, and G) encode the twenty amino acids that are the builiding blocks of proteins, as well as signals for start and stop of the three base codon translation into amino acids. In the cell, the code within the genomic DNA is copied to messenger RNA, where the nucleotide thymine (T) is replaced with uracil (U). In artifical systems, the messenger RNA can be obtained from cells and converted into DNA as a copy of the RNA message, called cDNA. Genomic DNA contains additional signals that are not part of the coding sequence.

The codon table below shows the code for translation of DNA triplets into amino acids.

CODON TABLE

First	T/U	C	A	G	Last
T/U	Phe	Ser	Tyr	Cys	T/U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	**Stop** (Ochre)	**Stop** (Umber)	A
	Leu	Ser	**Stop** (Amber)	Trp	G
C	Leu	Pro	His	Arg	T/U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G

A	Ile	Thr	Asn	Ser	T/U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	T/U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Sequencing

DNA can be sequenced from a variety of cell samples, including blood, hair, nail clippings, cheek cells, and the like. Known methods include didioxy sequencing, PCR sequencing, and the like. Random pieces of DNA extracted from cells may be sequenced, or specific portions may be isolated and sequenced, for example by targeted PCR amplification of a particular region of the DNA. Once the sample has been sequenced, it can be stored, for example, in an electronic data carrier, for transmission to a DNA transcriber for use in generating music according to the present invention.

Music

A musical composition is a collection of music notes that can include a melody, harmony, tempo, rhythm, volume, and the like. Depending upon the algorithm applied to a data stream, synthetic music can be generated in a variety of ways from simple to complex. In the simplest method, each of the DNA nucleotides A, C, T, and G can be assigned a specific musical note. The data stream of nucleotides can then be transcribed into a data stream of musical notes. Alternatively, the nucleotide sequence can first be decoded to an amino acid sequence, whereby the twenty amino acids can be each assigned to a specific musical note. Variations can be used to produce chords, to specify rhythms, tone, volume, to generate melodic music and harmonic music, and the like.

As shown in Figure 1, according to the present invention, a data stream of nucleotides of a DNA sequence is provided to a decoder. The decoder detects specific nucleotide sequences that may be recognition signals, enzyme recognition signals, promoter sequences, and the like. The decoder may also detect a coding sequence, for example a sequence having an open reading frame with start and stop signals, and intervening codon sequences. The DNA sequence received by the decoder is processed according to one set of recognition sequences and transcribed by a melodic sequence generator into a melodic sequence in response to the received DNA sequence. The decoder further processes the received DNA sequence into an amino acid sequence according to the identified triplet codon series, and conveys information related to the determined amino acids in a signal. In one embodiment, the harmonic sequence generator determines a series of chords according to the determined amino acid. The harmonic information can be used by the melodic sequence generator to develop a melodic sequence. Further, the harmonic sequence generator can identify chemical proerties that are associated with the determined amino acids. The melody and harmony generated are then processed into a music signal by a music signal generator.

As shown in Figure 2, in one embodiment, a melodic sequence is produced from the decoded DNA sequence by assigning each nucleotide to a chord position converter. The chord information is adjusted by an internal adjuster to produce a series of notes to generate a melodic sequence. The adjuster may use harmonic sequence information to assign notes in the melody as chord tones or passing tones.

As shown in Figure 3, in one embodiment, a harmonic sequence is produced from the decoded DNA sequence by classifying each amino acid identified from the DNA sequence. The classification can be according to a chemical property (such as "hydrophobic" or acidity) and/or simply by the amino acid name. The classification data is converted into chord types by a chord type selector that selects chord roots to produce a harmonic sequence. At the same time, the chord type and root selectors provide chord information that can be applied to the adjustment of data to obtain the melodic sequence, as shown in Figure 2.

As shown in Figure 4, the melodic sequence and the harmonic sequence are received by a music output module to produce the product music signal. The music signal may be an audio waveform that can be played on commercial music players or music command sequences (such as MIDI files or data streams) that can be used to control devices that generate music (such as musical synthesizers).

Products

The music signal generated from the genetic data can be used in a variety of consumer and industrial products and methods. For example, novelty products such as greeting cards, genetic music CDs, and the like can incorporate a person's individual music generated from their own sample of DNA. The specific DNA sequence can be provided to a company for generation of the genetic music. Alternatively, a sample containing the genetic material can be provided for sequencing and generating the music.

Useful products include individual identity analysis, for example, for security checking, paternity testing, and the like. The music generated by an individual sample can be compared with a control sample. An identity analyzer can be configured to provide an audible signal for a specific comparative result, for example, if the sample and the control differ, e.g., signaling an alarm in a security setting, or when they are the same, e.g., adding excitement to live television coverage of paternity determinations.

Clinical analyzers that compare sequences of patient samples with controls may be programmed to provide soothing melodies when the sequence is "normal" and to provide an audible, for example, discordant music when an "abnormal" sequence is detected. Such signals can provide a signal for the clinical technician to alert a physician to the difference in the sequence.

The above specification, examples, figures, and data provide a complete description of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

Example 1.

To exemplify the invention, a program was devised to analyze a DNA sequence and generate music according to the invention:

```
Private Sub WriteMidiFile(Melody)
  MsgBox "Procedure WriteMidiFile entered"
Dim bytearray(0 To 65535) As Byte
  bytearray(0) = Asc("M") 'header chunk 1
  bytearray(1) = Asc("T")
  bytearray(2) = Asc("h")
  bytearray(3) = Asc("d")
  bytearray(4) = 0
  bytearray(5) = 0
  bytearray(6) = 0
  bytearray(7) = 6       'length of file in bytes
  bytearray(8) = 0
  bytearray(9) = 1       'Format Type 1 Multitrack
  bytearray(10) = 0
  bytearray(11) = 2      'Number of tracks in this File
  bytearray(12) = 1
  bytearray(13) = 172    '? ticks per beat

  bytearray(14) = Asc("M")  'header chunk 2
  bytearray(15) = Asc("T")
  bytearray(16) = Asc("r")
  bytearray(17) = Asc("k")
  bytearray(18) = 0
  bytearray(19) = 0
  bytearray(20) = 0
  bytearray(21) = 26     'length of file in bytes
  bytearray(22) = 0      'separator

  bytearray(23) = 255    'Tempo setting command (3bytes)
  bytearray(24) = 5 * 16 + 1
  bytearray(25) = 3
  bytearray(26) = 7      'microseconds
  bytearray(27) = 10 * 16 + 1
  bytearray(28) = 32
  bytearray(29) = 0    'separator

  bytearray(30) = 255    'Time Signature command (3bytes)
  bytearray(31) = 5 * 16 + 8
  bytearray(32) = 4
  bytearray(33) = 3      'numerator
  bytearray(34) = 2      'denomenator
  bytearray(35) = 96     'Clocks / beat
  bytearray(36) = 8      '32nd notes / beat
  bytearray(37) = 0      'separator
```

```
  bytearray(38) = 255    '?Key Signature setting command (3bytes)
  bytearray(39) = 127
  bytearray(40) = 3
  bytearray(41) = 0      'flats
  bytearray(42) = 0      'sharps
  bytearray(43) = 65     '?
  bytearray(44) = 0      'separator

  bytearray(45) = 255      'End of Track command
  bytearray(46) = 2 * 16 + 15
  bytearray(47) = 0

  bytearray(48) = Asc("M")   'header chunk 3
  bytearray(49) = Asc("T")
  bytearray(50) = Asc("r")
  bytearray(51) = Asc("k")
  bytearray(52) = 0
  bytearray(53) = 0
  bytearray(54) = 0
  bytearray(55) = 41     'length of file in bytes
  bytearray(56) = 0      'status
'discontinuity


 bytearray(66) = 9 * 16  'Note on, Middle C, vel. 64
 bytearray(67) = 60
 bytearray(68) = 64
 bytearray(69) = 128 + 3   'delay 432 ticks
 bytearray(70) = 3 * 16
 bytearray(71) = 128     'Note off, Middle C, vel. 64
 bytearray(72) = 60
 bytearray(73) = 64
 bytearray(74) = 3 * 16 'delay 432 ticks

 bytearray(75) = 9 * 16  'Note on, Middle C, vel. 64
 bytearray(76) = 60
 bytearray(77) = 64
 bytearray(78) = 128 + 3   'delay 432 ticks
 bytearray(79) = 3 * 16
 bytearray(80) = 128     'Note off, Middle C, vel. 64
 bytearray(81) = 60
 bytearray(82) = 64
 bytearray(83) = 3 * 16 'delay 432 ticks

 bytearray(84) = 9 * 16  'Note on, Middle C, vel. 64
 bytearray(85) = 60
 bytearray(86) = 64
 bytearray(87) = 128 + 3   'delay 432 ticks
 bytearray(88) = 3 * 16
 bytearray(89) = 128     'Note off, Middle C, vel. 64
 bytearray(90) = 60
 bytearray(91) = 64
 bytearray(92) = 3 * 16 'delay 432 ticks
```

```
  bytearray(93) = 9 * 16  'Note on, Middle C, vel. 64
  bytearray(94) = 60
  bytearray(95) = 64
  bytearray(96) = 128 + 3   'delay 432 ticks
  bytearray(97) = 3 * 16
  bytearray(98) = 128     'Note off, Middle C, vel. 64
  bytearray(99) = 60
  bytearray(100) = 64


  bytearray(101) = 10 * 16 + 13 'delay 5808 ticks
  bytearray(102) = 3 * 16
  bytearray(103) = 255     'End of Track command
  bytearray(104) = 2 * 16 + 15
  bytearray(105) = 0
'    put(#1, 10 * 16 + 13) & put(#1, 3 * 16) 'Final ticks 5808
'    put(#1, 255) & put(#1, 2 * 16 + 15) & put(#1, 0) 'End of Track command
'
 'Print Chr(ByteArray(0)), Chr(ByteArray(1)), Chr(ByteArray(2)), Chr(ByteArray(3))
 'Print ByteArray(0), (1), ByteArray(2), ByteArray(3)
 ''Writebinaryfile bytearray(), 105
End Sub

Sub Writebinaryfile(track() As Byte, termcount As Integer)
  CommonDialog1.ShowSave
  Open CommonDialog1.FileName For Binary As 1
    For i = 0 To termcount
      Put 1, , track(i)
    Next i
  Close 1
End Sub



Private Sub cmdReadBinFile_Click()

CommonDialog1.ShowOpen
Open CommonDialog1.FileName For Binary As 2
  n = LOF(2) 'the whole length of this file
  i = 1



Dim a(0 To 5000) As Byte
Do While (i <= n)
  b = Input(1, #2)
  ''If i = 1 Then MsgBox ("M=" & Asc("M") & " " & i & " " & b & " " & Asc(b))
  'If i = 2 Then MsgBox ("T=" & Asc("T") & " " & i & " " & b & " " & Asc(b))
  'If i = 3 Then MsgBox ("h=" & Asc("h") & " " & i & " " & b & " " & Asc(b))
  'If i = 4 Then MsgBox ("d=" & Asc("d") & " " & i & " " & b & " " & Asc(b))
  'If i = 5 Then MsgBox (" =" & Asc(" ") & " " & i & " " & b & " " & Asc(b))
  If b <> "" Then a(i) = Asc(b)
  int1 = Asc(b)
  'Print i & " " & int1 & "  "
  If i = 56 Then
    temp = int1
```

```
    End If
    i = i + 1
  Loop

    'MsgBox ("M=" & Asc("M") & " " & i & " " & ChrB(a(1)))
    '  MsgBox ("h=" & Asc("h") & " " & i & " " & a(2) & " " & ChrB(a(2)))
    '  MsgBox ("d=" & Asc("d") & " " & i & " " & a(3) & " " & ChrB(a(3)))
    '  MsgBox (" =" & Asc(" ") & " " & i & " " & a(4) & " " & ChrB(a(4)))
  i = n
  inputfilelen = i
Close 2

MsgBox (CommonDialog1.FileName & " is " & n & " long.  n = " & n & "i = " & i)

CommonDialog1.ShowSave
Open CommonDialog1.FileName For Output As 3
For i = 1 To inputfilelen
  Print #3, i; Tab(10); (a(i))
Next i

Close 3


End Sub

Private Sub cmdWriteMidiTest_Click()
Static Melody
Dim note As Byte
Dim track(0 To 2 ^ 16) As Byte
Dim J As Integer

t = 0

track(0) = Asc("M") 'header chunk 1
  track(1) = Asc("T")
  track(2) = Asc("h")
  track(3) = Asc("d")
  track(4) = 0
  track(5) = 0
  track(6) = 0
  track(7) = 6       'length of file in bytes
  track(8) = 0
  track(9) = 1       'Format Type 1 Multitrack
  track(10) = 0
  track(11) = 2      'Number of tracks in this File
  track(12) = 1
  track(13) = 128    '? ticks per beat

  track(14) = Asc("M")  'header chunk 2
  track(15) = Asc("T")
  track(16) = Asc("r")
  track(17) = Asc("k")
  track(18) = 0
  track(19) = 0
  track(20) = 0

  track(21) = 26     'length of file in bytes
  track(22) = 0      'separator
```

```
  track(23) = 255   'Time Signature command (3bytes)
  track(24) = 5 * 16 + 8
  track(25) = 4
  track(26) = 3     'numerator
  track(27) = 2     'denomenator
  track(28) = 96    'Clocks / beat
  track(29) = 8     '32nd notes / beat
  track(30) = 0     'separator

  track(31) = 255   '?Key Signature setting command (3bytes)
  track(32) = 127
  track(33) = 3
  track(34) = 0     'flats
  track(35) = 0     'sharps
  track(36) = 65    '?
  track(37) = 0     'separator

  track(38) = 255   'Tempo setting command (3bytes)
  track(39) = 5 * 16 + 1
  track(40) = 3
  track(41) = 7     'microseconds
  track(42) = 10 * 16 + 1
  track(43) = 32
  track(44) = 0   'separator

  track(45) = 255     'End of Track command
  track(46) = 2 * 16 + 15
  track(47) = 0

  track(48) = Asc("M")  'header chunk 3
  track(49) = Asc("T")
  track(50) = Asc("r")
  track(51) = Asc("k")
  track(52) = 0
  track(53) = 0
  track(54) = 0
  track(55) = 133    'length of file in bytes
  track(56) = 0     'status

  track(57) = 255 'command
  track(58) = 89
  track(59) = 2
  track(60) = 0
  track(61) = 0
  track(62) = 0
DNAString = "ATGATGATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGCGAAGGTGAACGTGGATGAA'
'DNAString = "TTTTTCTTATTGTCTTCCTCATCGTATTACTAATAGTGTTGCTGATGG"
'DNAString = "CTTCTCCTACTGCCTCCCCCACCGCATCACCAACAGCGTCGCCGACGG"
'DNAString = "ATTATCATAATGACTACCACAACGAATAACAAAAAGAGTAGCAGAAGG"
'DNAString = "GTTGTCGTAGTGGCTGCCGCAGCGGATGACGAAGAGGGTGGCGGAGGG"
'DNAString = "ATGTTT"

DNAlen = Len(DNAString)
i = 1
J = 63
'Do While i <= DNAlen
'  nuke = UCase(Mid(DNAString, i, 1))
'  Select Case nuke
'    Case "C"
'      note = 60
'    Case "A"
'      note = 57
'    Case "T"
'      note = 62
'    Case "G"
'      note = 67
'    Case "U"
```

```
'       MsgBox "Warning: Viral DNA detected"
'       note = 66
'     Case Else
'       MsgBox "Warning: Alien DNA detected"
'       note = 99
'   End Select

'   track(J) = 9 * 16 'Note on, Middle C, vel. 64
'   track(J + 1) = note
'   track(J + 2) = 64
'   track(J + 3) = 128 + 3 'delay 432 ticks
'   track(J + 4) = 3 * 16
'   track(J + 5) = 128    'Note off, Middle C, vel. 64
'   track(J + 6) = note
'   track(J + 7) = 64
'   track(J + 8) = 3 * 16 'delay 432 ticks
'   J = J + 9
'
'   i = i + 1
   'reply = MsgBox(nuke & note & " " & i & " " & Len(Melody), 51)
   'If reply = vbCancel Then i = DNAlen + 1
'Loop

  ' track(J - 1) = 10 * 16 + 13 'delay 5808 ticks
'track(J) = 3 * 16
  ' track(J + 1) = 255    'End of Track command
'   track(J + 2) = 2 * 16 + 15
'   track(J + 3) = 0

'reply = MsgBox("Do you want a test MIDI file?", 51)
'If reply = vbYes Then WriteMidiFile (Melody)
'i = i - 1
'track(63) = (((i * 9) + 5) \ 128)+128
'track(64) = (i * 9 + 5) Mod 128
'J = J + 3

croot = 60
voice1 = 0
voice2 = 0
cronvol = 100
cdonvol = 100
mdonvol = 100
croffvol = 64
cdoffvol = 64
mdoffvol = 64

k = 1
Do While k <= DNAlen
   nuke1 = UCase(Mid(DNAString, k, 1))
   nuke2 = UCase(Mid(DNAString, k + 1, 1))
   nuke3 = UCase(Mid(DNAString, k + 2, 1))
  'decode aa from 3 nukes; aa's are numbered alphabetically
  'aa= amino acid; nuke = nucleotide;
```

```
Select Case nuke1
  Case "T"
    Select Case nuke2
      Case "T"
        If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 14
        Else
          aa = 11
        End If
      Case "C"
          aa = 16
      Case "A"
        If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 19
        Else
          aa = 21
        End If
      Case "G"
        If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 5
        ElseIf nuke3 = "A" Then
          aa = 21
        Else
          aa = 18
        End If
    End Select
  Case "C"
    Select Case nuke2
      Case "T"
          aa = 11
      Case "C"
          aa = 15
      Case "A"
         If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 9
        Else
          aa = 6
        End If
      Case "G"
          aa = 2
    End Select
  Case "A"
    Select Case nuke2
      Case "T"
        If nuke3 = "G" Then
          aa = 13
        Else
          aa = 10
        End If
      Case "C"
          aa = 17
      Case "A"
        If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 3
        Else
          aa = 12
```

```
        End If
      Case "G"
        If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 16
        Else
          aa = 2
        End If
    End Select

  Case "G"
    Select Case nuke2
      Case "T"
          aa = 20
      Case "C"
          aa = 1
      Case "A"
         If (nuke3 = "T" Or nuke3 = "C") Then
          aa = 4
        Else
          aa = 7
        End If
      Case "G"
          aa = 8
    End Select
  Case "U"
    MsgBox "Warning: Viral DNA detected, SKIPPING U"
    note = 66
  Case Else
    MsgBox "Warning: Alien DNA detected, SKIPPING ?"
    note = 99
End Select



'MsgBox nuke1 & nuke2 & nuke3 & " = " & aa
k = k + 3
'assign chord type based on "hydrophobic" to "hydrophylic"
'ie, according to chemical property
Select Case aa
  Case 14, 19, 18
    ct = 1 'ct = chordtype
  Case 8, 1, 20, 11, 10
    ct = 2
  Case 16, 17
    ct = 3
  Case 15
    ct = 4
  Case 5, 13
    ct = 5 'ct = chordtype
  Case 3, 6
    ct = 6
  Case 12, 2, 9
    ct = 7
  Case 4, 7
    ct = 8
  End Select
'assume root = 0, chord tones in semitones
'define chord types (chord tones are adj. based on aa type)
```

```
Select Case ct
  Case 1 'augmented chord type
    ct1 = 4
    ct2 = 8
    ct3 = 12
  Case 2 'maj. 7th
    ct1 = 4
    ct2 = 7
    ct3 = 11
  Case 3 '13th chord
    ct1 = 4
    ct2 = 7
    ct3 = 9
  Case 4  '9th chord
    ct1 = 4
    ct2 = 7
    ct3 = 14
  Case 5  'sus 4th chord
    ct1 = 5
    ct2 = 7
    ct3 = 12
  Case 6  'dominant 7th
    ct1 = 4
    ct2 = 7
    ct3 = 10
  Case 7  'half diminished
    ct1 = 3
    ct2 = 6
    ct3 = 10
  Case 8  'diminished
    ct1 = 3
    ct2 = 6
    ct3 = 9
End Select
'adj. root based on chem. property
'in this example, the interval is determined by the degree of
'"hydrophobia".  the intervals are diatonic (i.e., in a modal scale)
'other scales can be used

Select Case ct
  Case 1 'augmented chord type
    croot = croot + 7
  Case 2 'maj. 7th
    croot = croot + 5
  Case 3 '13th chord
     croot = croot + 4

  Case 4  '9th chord
       croot = croot + 2
  Case 5  'sus 4th chord
   croot = croot
  Case 6  'dominant 7th
         croot = croot - 1
  Case 7  'half diminished
    croot = croot - 2
  Case 8  'diminished
    croot = croot - 3
End Select

'clip if necessary to keep in range; can be more intelligent
   If croot > 67 Then croot = croot - 12
   If croot < 36 Then croot = croot + 12
```

```
'assign melody based on nukes and chord tones (which are
' determined from aa's)
  Select Case nuke1
    Case "C"
      note1 = croot + 12 + 12
    Case "A"
      note1 = croot + ct1 + 12
    Case "T"
      note1 = croot + ct2 + 12
    Case "G"
      note1 = croot + ct3 + 12
  End Select
  Select Case nuke2
    Case "C"
      note2 = croot + 12 + 12
    Case "A"
      note2 = croot + ct1 + 12
    Case "T"
      note2 = croot + ct2 + 12
    Case "G"
      note2 = croot + ct3 + 12
  End Select
  Select Case nuke3
    Case "C"
      note3 = croot + 12 + 12
    Case "A"
      note3 = croot + ct1 + 12
    Case "T"
      note3 = croot + ct2 + 12
    Case "G"
      note3 = croot + ct3 + 12
  End Select
  'generate midi stuff

  track(J) = 9 * 16 'Note on, croot, vel. 64
  track(J + 1) = croot - 12
  track(J + 2) = cronvol
  track(J + 3) = 0
  track(J + 4) = 9 * 16 + voice1 'Note on, ct1, vel. 64
  track(J + 5) = croot + ct1
  track(J + 6) = cdonvol
  track(J + 7) = 0
  track(J + 8) = 9 * 16 + voice1 'Note on, ct2 C, vel. 64
  track(J + 9) = croot + ct2
  track(J + 10) = cdonvol
  track(J + 11) = 0
  track(J + 12) = 9 * 16 + voice1 'Note on, ct3, vel. 64
  track(J + 13) = croot + ct3
  track(J + 14) = cdonvol
  track(J + 15) = 0
  track(J + 16) = 9 * 16 + voice2 'Note on, melody, vel 72 432 ticks
  track(J + 17) = note1
  track(J + 18) = mdonvol
  track(J + 19) = 130 'delay 432 ticks+48
```

```
                track(J + 20) = 0
                track(J + 21) = 128 'Note off, croot, vel. 64
                track(J + 22) = croot - 12
                track(J + 23) = croffvol
                track(J + 24) = 0
                track(J + 25) = 128 + voice1 'Note off, ct1, vel. 64
                track(J + 26) = croot + ct1
                track(J + 27) = cdoffvol
                track(J + 28) = 0
                track(J + 29) = 128 + voice1 'Note off, ct2, vel. 64
                track(J + 30) = croot + ct2
                track(J + 31) = cdoffvol
                track(J + 32) = 0
                track(J + 33) = 128 + voice1 'Note off, ct3, vel. 64
                track(J + 34) = croot + ct2
                track(J + 35) = cdoffvol
                track(J + 36) = 0
                track(J + 37) = 128 + voice2 'note off melody
                track(J + 38) = note1
                track(J + 39) = mdoffvol
                track(J + 40) = 129 'delay
                track(J + 41) = 0
                track(J + 42) = 9 * 16 + voice2 'Note on, melody, vel 72 432 ticks
                track(J + 43) = note2
                track(J + 44) = mdonvol
                track(J + 45) = 130 'delay n ticks
                track(J + 46) = 0
                track(J + 47) = 128 + voice2 'note off melody
                track(J + 48) = note2
                track(J + 49) = mdoffvol
                track(J + 50) = 129 'delay
                track(J + 51) = 0
                track(J + 52) = 9 * 16 + voice2 'Note on, melody, vel 72 432 ticks
                track(J + 53) = note3
                track(J + 54) = mdonvol
                track(J + 55) = 130 'delay
                track(J + 56) = 0
                track(J + 57) = 128 + voice2 'note off melody
                track(J + 58) = note3
                track(J + 59) = mdoffvol
                track(J + 60) = 129 'delay
                track(J + 61) = 0
                loopsize = 62

                  J = J + loopsize
Loop

  track(J - 2) = 0
  track(J - 1) = 255 'End of Track command
  track(J) = 2 * 16 + 15
  track(J + 1) = 0
  J = J + 1

  loops = k \ 3
  track(54) = ((7 + loops * loopsize) + 2) \ 256
  temp = 7 + loops * loopsize + 2
  track(55) = (7 + loops * loopsize + 2) Mod 256
  Dim varsize As Integer

  MsgBox loops

Writebinaryfile track(), J  'debug -> J   End Sub

                                          Private Sub Command2_Click()

                                          End Sub
```